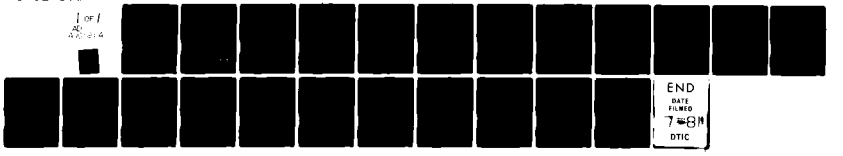
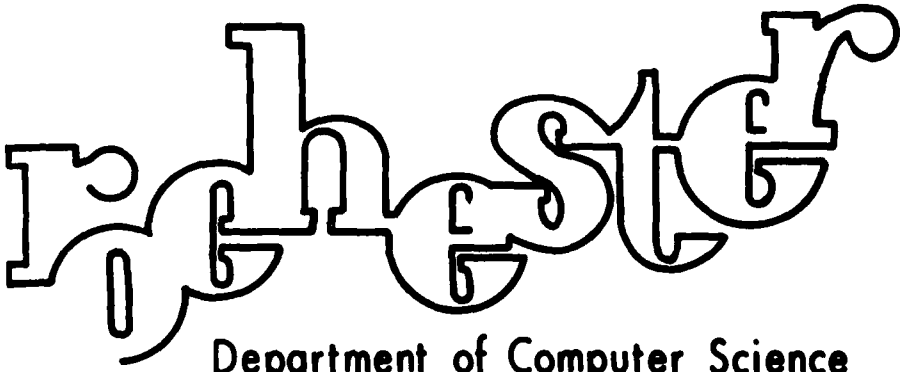
AD-A101 214 ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE F/G 12/1
ANALYSIS OF 'DOT PRODUCT SPACE' SHAPE DESCRIPTIONS,(U)
JUN 80 K R SLOAN N00014-78-C-0164
UNCLASSIFIED TR-74 NL



END
DATE FILMED
7-81
DTIC

(12)

LEVEL II



Department of Computer Science
University of Rochester
Rochester, New York 14627

LEVEL II

12

12 93

6 Analysis of 'Dot Product Space' Shape Descriptions

10 K.R. Sloan, Jr.
Computer Science Department
The University of Rochester
Rochester, NY 14627

TR74
11 Jun 80

14 TR-74

Abstract

A convenient representation for blob-like figures in an image consists of the orientation, length, and width of a bounding rectangle. One fast algorithm for producing such a bounding rectangle is based upon a dot product space. The analysis of the dot product space shape representation is improved to handle certain pathological cases, and it is shown how to generalize this analysis to accommodate different criteria for the goodness of the representation.

The preparation of this paper was supported in part by the Defense Advanced Research Projects Agency, monitored by the ONR, under Contract No. N00014-78-C-0164.

I. Introduction

In many computer vision applications, it is necessary to describe the gross shape of blob-like figures. These figures may be, for example, cells in histological preparations [3], elements in a natural texture [7], or flaws in manufactured items [5].

A convenient representation for these blob-like figures in an image consists of the *orientation*, *length*, and *width* of a *bounding rectangle*. These descriptors can also be used to describe an *ellipse*. In fact, the bounding rectangle descriptors are most appropriate when blob-like really means *close to elliptical*.

One fast algorithm for producing such a bounding rectangle is based upon a *dot product space* [5,2]. This method is similar to Hough-like techniques [12] in that local evidence about the nature of the shape (boundary elements) is projected into a transform space, where it is relatively easy to make a global judgment. Such methods can be very robust in the presence of occlusions and incomplete boundary information. They may also lend themselves to very fast parallel or vector-machine implementations.

As with Hough-like techniques, the dot product space method consists of two basic phases. First, all of the boundary points (the local evidence) are projected into a transform space, which has been tiled with a set of bins. Then, the contents of these transform space bins are analyzed and the "answer" is extracted. Such transform methods are particularly appropriate when the analysis of the transform space involves a simple, well-understood operation. Most of the time, simple peak finding is all that is required.

The implementation described by Firschein et al. [5] accomplished both the transformation and analysis operations with the use of a vector-machine. Their final analysis phase is direct and simple, but behaves poorly (in the sense that the resulting descriptions are poor) for certain pathological cases.

In this paper, the analysis of the dot product space representation of a shape is improved to handle some pathological cases in a reasonable way, and it is shown how to generalize this analysis phase to accommodate different criteria for the *goodness* of the representation.

Section II motivates the use of bounding rectangles as descriptors. The dot product space is described in Section III. Section IV presents the results of an experimental study of the relative effectiveness of four methods of deriving bounding rectangles from the dot product space representation of a figure.

II. Bounding Rectangles

Shape is an important attribute of individual figures in an image. Fortunately, it is often enough to treat a figure as a blob, with an overall *orientation*, *length*, and *width*. These descriptors, along with *position*, completely specify an arbitrary rectangle. In applications where the spatial extent of the figure is the feature of interest, it is usual to describe the blob by giving a *bounding rectangle*, in which the figure is inscribed.

<See Figure 1>

For a given figure, there are many such bounding rectangles, some better than others. The easiest bounding rectangle to construct is defined by the minimum and maximum extent of the figure in the X and Y directions. Finding this rectangle corresponds to projecting the figure onto the X and Y axes of the co-ordinate system and noting the range of the image of the figure on each axis. A better bounding rectangle can be generated, at much greater computational cost, by calculating the orientation of the figure (using the eigenvectors of the scatter matrix of the region points [4,13]). If the application demands that the *length* descriptor be equal to the distance between the two most distant boundary points (*diameter*) then yet another bounding rectangle may result.

Note that for truly elliptical blobs, these last two criteria will be satisfied by the same bounding rectangle, but that they will disagree when the figure deviates significantly from an ellipse. Most reasonable criteria will agree with these last two for elliptical blobs, but may differ for others. For example, we may wish to minimize area, minimize or maximize aspect ratio, or minimize width.

<See Figure 2>

Strips

Bounding rectangles may be useful for open curves as well as for the closed boundaries of blob-like features. The *Strip Tree* representation for digital curves (both open and closed) is based on bounding rectangle (*Strip*) approximations to the curve, where the orientation of the strip is determined by the straight line connecting the first and last points in the curve, and the length and width are determined by the extent of the curve points parallel and perpendicular to that orientation [1]. The bounding rectangle generation methods described here may be useful in obtaining good initial approximations for closed curves in that representation.

Ellipses

From the above discussion, it may appear that bounding rectangles are most useful when the figures are elliptical. In that case, why not simply use an approximating ellipse rather than a bounding rectangle? In many ways, the two methods are identical, and it is easy to generate either one from the descriptors: orientation, length, width. The difference lies mainly in what the description asserts about the original figure, and in how the description is derived. The bounding rectangle point of view is appropriate when gross spatial extent is the property of interest. Elliptical approximations are useful when the interesting properties have to do with moments of inertia of the blobs [7,14].

As with enclosing rectangles, there are several measures of the *goodness* of an elliptical approximation to an arbitrary blob. For example, one may wish to preserve *area*, *orientation*, and *aspect ratio* (length/width) rather than require that the approximating ellipse strictly enclose the original figure.

<See Figure 3>

III. Dot Product Space

The dot product space method of shape description depends on the following three observations:

a) the *length* of a figure along a particular orientation is exactly the extent of the image of the figure, projected into a line at that orientation.
b) the required projection operation is simply the dot product of the vector representing an individual point with the unit vector at the desired orientation.
c) in most applications, we can afford to *approximate* the orientation which yields the maximum length (and as a result the value of the maximum length).

Once the required precision of the orientation calculation is known, a set of *unit vectors* can be chosen to span the range [0,180) degrees at that precision. Associated with each unit vector are two *accumulators*, which are used to store the two extremes of the figure's projection.

The maximum length of the blob is now simply the maximum value of the extent of a blob's projection as a function of orientation. The orientation of the blob is the orientation which sees this maximal projection. Furthermore, the width of the bounding rectangle at this orientation is the extent of the projection of the blob as seen at an orientation offset 90 degrees from the major axis. All of this is simple to compute, given the set of accumulators tiling the dot product space.

<See Figure 4>

The idea of calculating extent along a particular axis by projecting points on that axis is discussed briefly by Freeman [6]. The construction of a regular k-gon of viewing lines, followed by the calculation of the extents of the images of a set of points is analyzed in detail by Brown [2].

Firschein et al. [5] discuss an Array Processor formulation of the above process in the context of automatic inspection of radiographs of artillery shells. It appears that their application required that the length of the resulting bounding rectangle be the maximum distance between any two boundary points. As they note, this choice of criterion for the goodness of the bounding rectangle behaves well for elliptical blobs, but produces very bad results for other shapes.

In particular, consider a rectangle. Clearly, this figure can be well approximated (!) by a bounding rectangle. However, the method described above yields a particularly poor approximation. Moreover, for any original figure, if for some reason the approximation procedure is repeated, this method produces an unbounded sequence of bounding rectangles, each one larger than, and at an orientation completely different from, the previous approximation. (This last objection disappears if the input to each iteration is an ellipse generated from the orientation, length, and width descriptors.)

Generalizing the Analysis

The problem with the above method is that the measure of goodness of resulting bounding rectangles is that *length* equal *diameter*. This means that the length of a rectangular figure is taken to be the *diagonal*, and hence, the orientation of the approximation is guaranteed to be different than the orientation of the original rectangular figure.

Some applications may, in fact, require the *maximum length* approximation. In other applications, however, other measures (*width*, *area*, *aspect ratio*) may be more appropriate. Fortunately, the dot product space intermediate representation of the shape of the original figure can also be used to produce an optimal bounding rectangle for each of these measures.

In the intermediate dot product space representation of a figure, we have the width of the figure as seen in orthogonal projection as a function of orientation:

$$W(\Theta).$$

The *diameter* computation proceeds by calculating the maximum value of this function. The *maximum length bounding rectangle* computation then takes advantage of the fact that the width of this rectangle is given by:

$$W(\Theta_{Max} \pm 90 \text{ degrees})$$

Now, if the bounding rectangle of minimal area is required, all that is necessary is to combine the above two ideas. Rather than seek the maximum value of W, we instead calculate the minimum value of:

$$A(\Theta) = W(\Theta)*W(\Theta+90) \text{ for } 0 \leq \Theta < 90 \text{ degrees.}$$

Similar calculations can be performed for any measure which takes into account *length*, *width*, and *orientation* of the bounding rectangle. Since these parameters completely specify the bounding rectangle, the dot product space intermediate representation can be used to generate an optimal bounding rectangle whenever the measure of goodness depends only on the bounding rectangle itself.

This intermediate representation is similar to those described in Sloan [11] and Maruyama [8], and is useful independent of the generation of bounding rectangles and approximating ellipses. The extents of the k projections give a k-gon approximation to the convex hull of p points in O(kp) time.

IV. Experimental Results

There are two distinct reasons for generating bounding rectangles (or approximating ellipses) from a set of points. The first is to describe the actual set of observed points themselves. From this point of view, the previous section tells us that we can select arbitrary criteria for choosing a bounding rectangle. As long as the criteria are functions of the *length*, *width*, and *orientation* of the bounding rectangle, we can provide effective procedures which will produce the best such bounding rectangle from an analysis of $W(\Theta)$.

The second point of view is that the observed points represent evidence for some figure, and that the descriptors (*length*, *width*, and *orientation*) should describe that figure. This is often the case in computer vision applications.

With this second point of view in mind, it is possible to experimentally determine the relative effectiveness of different evaluation criteria. The two experiments presented below involve generating data points from a known figure, extracting bounding rectangles according to four different criteria, and calculating the range of parameter values as a function of the number of data points.

Ellipses

In Experiment I, data points were generated from the boundary of an ellipse with:

length = 100.0
width = 60.0
orientation = 30 degrees.

Using the pseudo-random number generator supplied by the SAIL run-time routines [10], sequences of numbers, r_i, in the range [0.0,1.0] were generated. The co-ordinates of a point on the boundary of an ellipse with:

length = 2a
width = 2b

centered at the origin were calculated as:

$$x_i := a * COSD(r_i * 360.0)$$
$$y_i := b * SIND(r_i * 360.0)$$

These points were then rotated about the origin by 30 degrees and translated by an arbitrary (fixed) vector.

The number of points in a sequence ranged from 3 to 50, with 100 sequences of each length. Each set of points was transformed into the dot product space and four bounding rectangles generated, meeting the respective criteria:

MAXIMIZE LENGTH
MINIMIZE WIDTH
MINIMIZE AREA
MAXIMIZE ASPECT RATIO

From the point of view of describing the observed data points, each of these rectangles was optimal with respect to its own criterion. From the point of view of describing the underlying figure, we can calculate the error in estimating the *length*, *width*, and *orientation* of the ellipse. The minimum, maximum, and mean errors in these estimates are shown, as functions of the number of data points, in Figure 5.

<See Figure 5>

Rectangles

Experiment I corresponds to one choice of underlying figure and data collection method. It appropriately models the case where:

a) the underlying figure is a perfect ellipse; and
b) data points are generated by boundary detection.

Experiment II changes these assumptins to:

a) the underlying figure is a rectangle; and
b) data points are generated by pixel classification (e.g., thresholding).

The co-ordinates of a point inside of a rectangle with:

$$\text{length} = 2a$$
$$\text{width} = 2b$$

centered at the origin were calculated as:

$$x_i = a * r_j$$
$$y_i = b * r_{j+1}$$

These points were rotated and translated exactly like the ellipse boundary points in Experiment I. Again, 100 trials each of 3 to 50 data points were generated, the four bounding rectangles chosen, and the range of error in the estimation of *length*, *width*, and *orientation* recorded. These data are shown in Figure 6.

<See Figure 6>

Discussion

Experiment I shows that the LENGTH criterion is generally superior to the others when the underlying figure is an ellipse. This superiority is most clearly seen in Figure 5c, which compares the errors in estimating orientation. The LENGTH method converges fastest, with ASPECT RATIO a close second. WIDTH is a clear third. The AREA method appears to be overly clever in fitting a bounding rectangle to the points. This generally involves placing data points near the corners of the rectangle. Since the underlying ellipse has no corners, this often produces bad estimates.

Figure 5a indicates that the LENGTH, WIDTH, and ASPECT RATIO criteria are nearly identical, with AREA clearly inferior. Again, in Figure 5b, AREA shows a wider range of errors. The range of errors for the WIDTH criterion is clipped at 0, since it is impossible to overestimate the width of an ellipse while minimizing the width of the bounding rectangle. Otherwise, the LENGTH and WIDTH errors are very similar. The ASPECT RATIO method also shows a tendency for the maximum error to be clipped at 0, but not as absolutely as for the WIDTH method.

Overall, when the underlying figure is an ellipse, the LENGTH criterion behaves very well, with ASPECT RATIO and WIDTH very close behind. AREA does a better job of

describing the actual data points than it does providing estimates of the parameters of an ellipse.

On the other hand, Experiment II demonstrates the problems that the LENGTH method has in describing figures with corners. Figure 6a indicates that the four methods show similar error ranges in estimating length. Figure 6b shows the expected clipping of the error in width estimation using the WIDTH and ASPECT RATIO criteria, and the occasional erratic behavior of the AREA method. It also shows that the LENGTH criterion is particularly bad in this situation. Similarly, Figure 6c shows a very broad error range for orientation estimates made by the LENGTH criterion. In fact, the LENGTH method is converging very nicely *to the two wrong answers* corresponding to the orientations of the rectangle's diagonals. ASPECT RATIO and WIDTH are clearly better, with AREA behaving about as well, except for the expected erratic behavior caused by its cleverness.

The strengths and weaknesses of these four methods are clearly demonstrated by these two experiments. The LENGTH method works very well for elliptical figures, but behaves poorly when the figures have corners. The AREA method, while appropriate for bottom-up descriptive applications, is occasionally very bad at providing estimates of the parameters of underlying figures. The WIDTH method provides biased estimates of the width of a figure, but is generally better behaved than the other two methods. The ASPECT RATIO method combines the strengths of the LENGTH and WIDTH methods, while avoiding their weaknesses.

V. Conclusion

Blob-like figures are important in many computer vision applications. A convenient representation for these blob-like figures in an image consists of the *orientation*, *length*, and *width* of a *bounding rectangle*, or *approximating ellipse*.

The *dot product space* representation of a figure provides a k-gon approximation to the convex hull of p points in 0(kp) time, from which bounding rectangles may be extracted. The best such rectangle, according to a given criterion, may be found in 0(k) time.

Of the four methods directly tested here, the MAXIMIZE LENGTH method performs well on elliptical figures, but MINIMIZE WIDTH is more robust. Combining length and width in the MINIMIZE AREA method provides good descriptions of the observations, but poor estimates of the underlying figure. Combining length and width in the MAXIMIZE ASPECT RATIO criterion provides a method which is both accurate and robust.

The methods described here are currently being used in a number of computer vision applications, and have proven useful even in a single serial processor environment. This is due mainly to the simplicity of the implementation, and the descriptive power of the dot product space representation. These factors more than offset the potential loss in speed when dealing with very small figures, with a small number of boundary points.

The amount of error introduced by using k discrete cells to tile the dot product space is easy to calculate (see [2] for an error analysis of the diameter approximation), and easy to control. The simplicity and inherent parallelism of the dot product space transform make it an ideal candidate for vector machine [5] or VLSI implementation.

References

[1] D.H. Ballard, Strip Trees: A Hierarchical Representation for Map Features, *Proceedings IEEE PRIP Conference* , Chicago, IL, IEEE CH1428-2C, 278-285, August 1979.

[2] K.Q. Brown, Geometric Transforms for Fast Geometric Algorithms, Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, December 1979.

[3] C.A. Curcio and K.R. Sloan, Jr., A Computer System Combining Mapping and Neuronal Morphology, TR79, Computer Science Department, University of Rochester, July, 1980.

[4] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis. New York: Wiley, 1973.

[5] O. Firschein, W. Eppler, and M.A. Fischler, A Fast Defect Measurement Algorithm and its Array Processor Mechanization, *Proceedings IEEE PRIP Conference*, Chicago, IL, IEEE CH1428-2C, 109-113, August 1979.

[6] H. Freeman and R. Shapira, Determining the Minimum-Area Encasing Rectangle for an Arbitrary Closed Curve, *CACM* 18, 7, July 1975.

[7] J. Maleson, C.M. Brown, and J.A. Feldman, Understanding Natural Texture, *Proceedings DARPA IU Workshop*, Palo Alto, CA, October 1977.

[8] K. Maruyama, A Study of Visual Shape Perception, UIUCDCS-R-72-533, Department of Computer Science, University of Illinois at Urbana-Champaign, 1972.

[9] F.P. Preparata and S.J. Hong, Convex Hulls of Finite Sets of Points in Two and Three Dimensions, *CACM* 20, 2, February 1977.

[10] J.F. Reiser, SAIL, Stanford Artificial Intelligence Laboratory Memo AIM-289; Computer Science Department Report No. STAN-CS-76-574, Stanford University, August 1976.

[11] K.R. Sloan, Jr., World Model Driven Recognition of Natural Scenes, Ph.D. Dissertation, Moore School of Electrical Engineering, University of Pennsylvania, June 1977.

[12] K.R. Sloan, Jr. and D.H. Ballard, Experience with the Generalized Hough Transform, *Proceedings: 5th IJCPR*, Miami Beach, December 1980.

[13] W.E. Snyder and D.A. Tang, Finding the Extrema of a Region, *IEEE Trans. PAMI* , May 1980.

[14] J.M. Tenenbaum, H.G. Barrow, and S.A. Weyl, Interactive Design of Texture Classifiers, in *Research in Interactive Scene Analysis*, Final Report for SRI Project 8721, SRI, Menlo Park, CA, 1975.

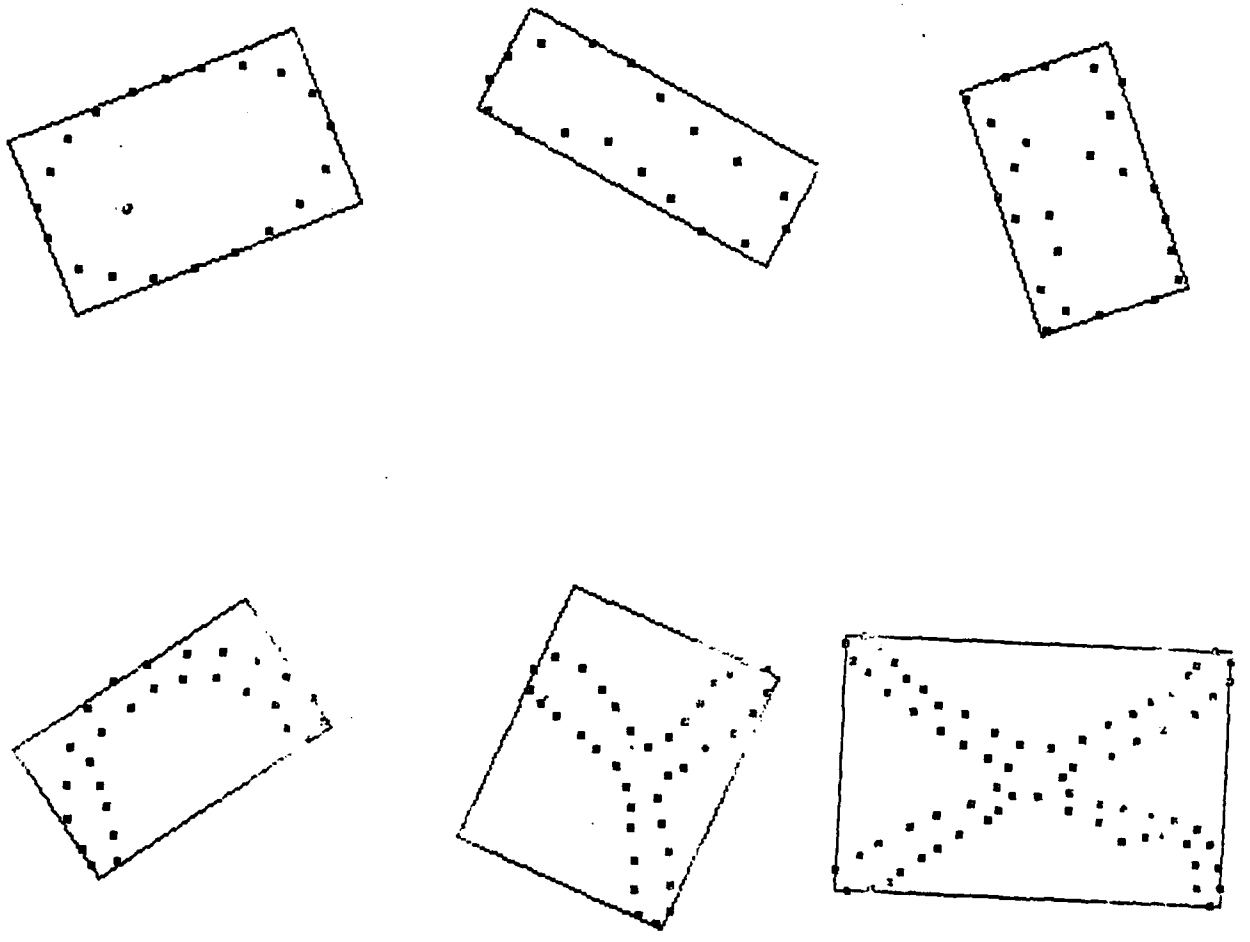
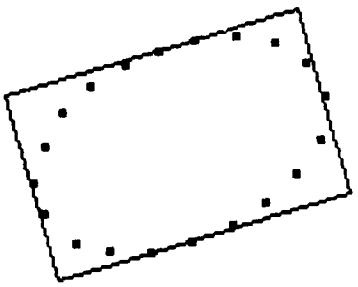
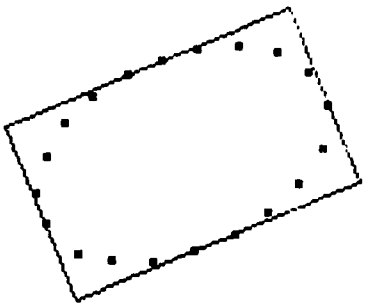


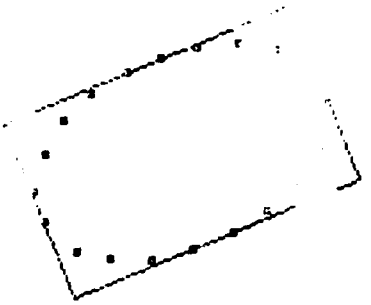
Fig. 1. a-c) Shapes for which a bounding rectangle is a good approximation;
d-f) Shapes for which a bounding rectangle is a bad approximation.



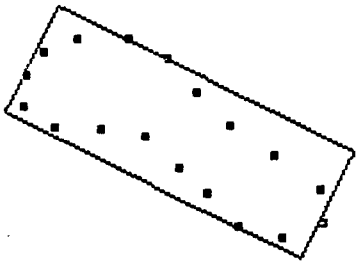
MAJOR AXIS = 118.14
MINOR AXIS = 74.57
ORIENTATION = 15.00
AREA = 8809.12



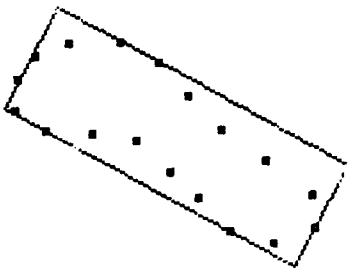
MAJOR AXIS = 118.10
MINOR AXIS = 73.16
ORIENTATION = 21.00
AREA = 8640.39



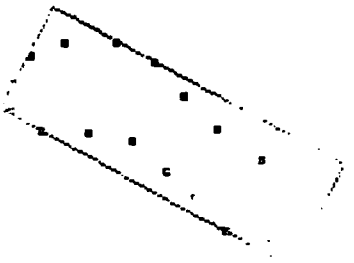
MAJOR AXIS = 118.10
MINOR AXIS = 73.16
ORIENTATION = 21.00
AREA = 8640.39



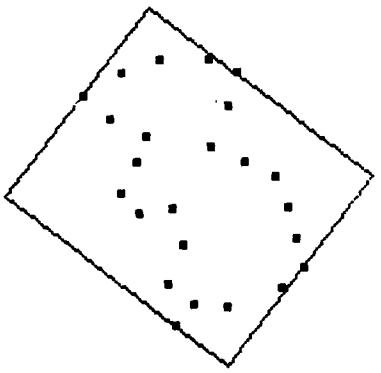
MAJOR AXIS = 128.80
MINOR AXIS = 45.98
ORIENTATION = 153.00
AREA = 5921.58



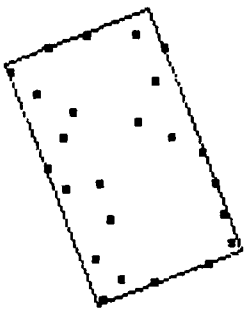
MAJOR AXIS = 128.70
MINOR AXIS = 44.07
ORIENTATION = 151.00
AREA = 5672.04



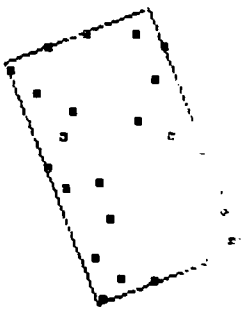
MAJOR AXIS = 128.70
MINOR AXIS = 44.07
ORIENTATION = 151.00
AREA = 5672.04



MAJOR AXIS = 108.23
MINOR AXIS = 92.62
ORIENTATION = 142.00
AREA = 10023.92

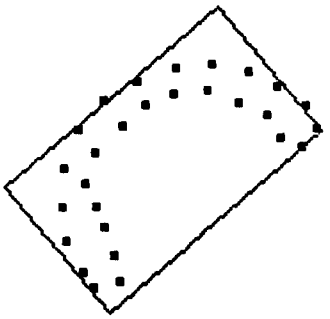


MAJOR AXIS = 99.80
MINOR AXIS = 59.46
ORIENTATION = 110.00
AREA = 5933.93

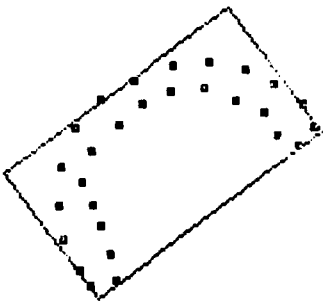


MAJOR AXIS = 99.80
MINOR AXIS = 59.46
ORIENTATION = 110.00
AREA = 5933.93

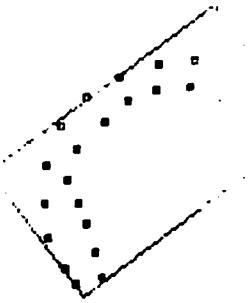
Fig. 2. a-f) Shapes with bounding rectangles satisfying:
i) maximum length; ii) minimum width; and iii) minimum area.



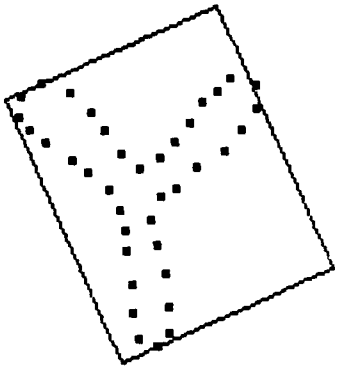
MAJOR AXIS = 108.07
MINOR AXIS = 63.37
ORIENTATION = 39.00
AREA = 6848.79



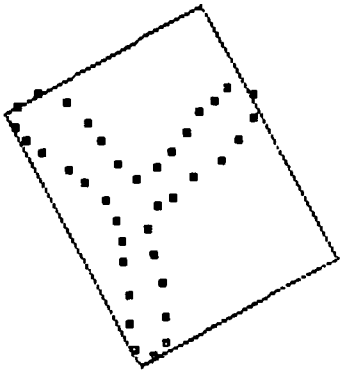
MAJOR AXIS = 107.81
MINOR AXIS = 59.97
ORIENTATION = 35.00
AREA = 6466.01



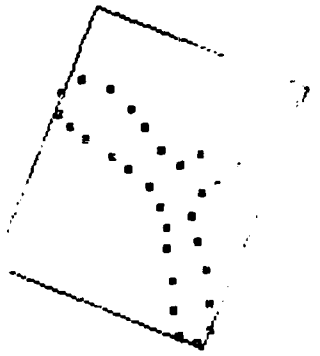
[illegible]OR AXIS = [illegible]1
[illegible]NOR AXIS = [illegible]7
[illegible]IENTATION = [illegible]0
[illegible]EA = [illegible]1



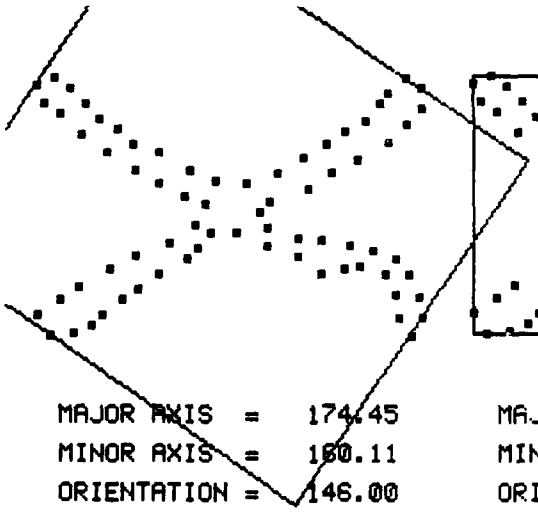
MAJOR AXIS = 111.23
MINOR AXIS = 89.05
ORIENTATION = 112.00
AREA = 9905.00



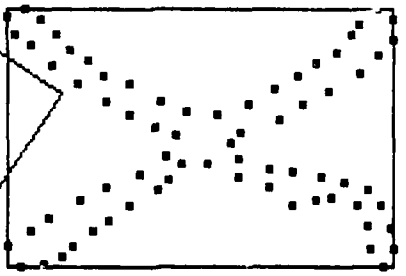
MAJOR AXIS = 110.84
MINOR AXIS = 86.81
ORIENTATION = 117.00
AREA = 9621.62



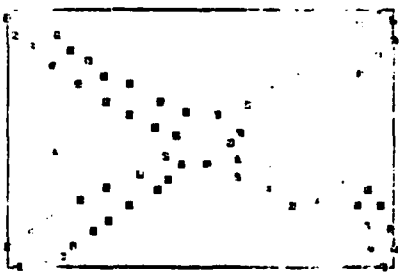
[illegible]OR AXIS = [illegible]7
[illegible]NOR AXIS = [illegible]7
[illegible]IENTATION = [illegible]0
[illegible]EA = [illegible]5



MAJOR AXIS = 174.45
MINOR AXIS = 160.11
ORIENTATION = 146.00
AREA = 27931.78

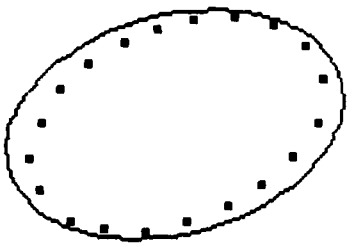


MAJOR AXIS = 149.00
MINOR AXIS = 100.00
ORIENTATION = .00
AREA = 14900.00

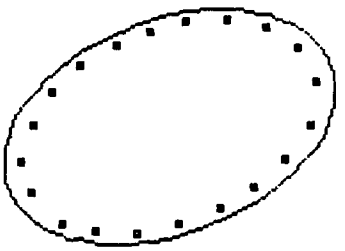


[illegible]JOR AXIS = [illegible]00
[illegible]NOR AXIS = [illegible]0
[illegible]IENTATION = [illegible]0
[illegible]EA = [illegible].00

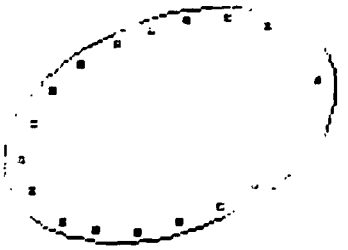
Fig. 2. Continued.



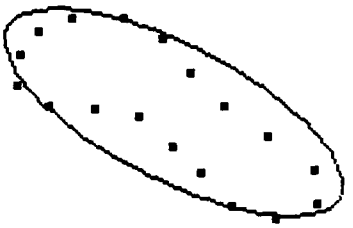
MAJOR AXIS = 118.14
MINOR AXIS = 74.57
ORIENTATION = 15.00
AREA = 8809.12



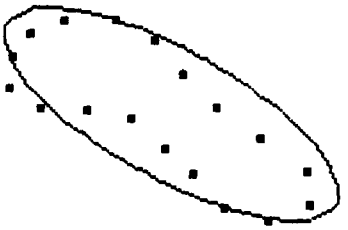
MAJOR AXIS = 118.10
MINOR AXIS = 73.16
ORIENTATION = 21.00
AREA = 8640.39



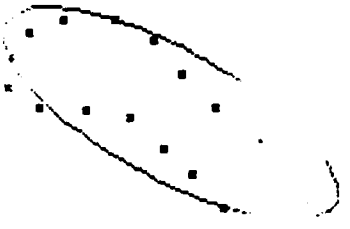
MAJOR AXIS = [illegible]
MINOR AXIS = [illegible]
ORIENTATION = [illegible]
AREA = [illegible]



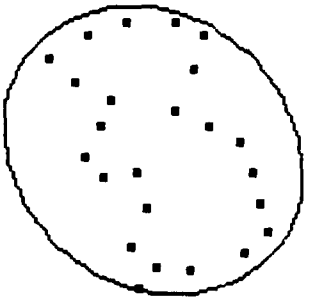
MAJOR AXIS = 128.80
MINOR AXIS = 45.98
ORIENTATION = 153.00
AREA = [illegible]



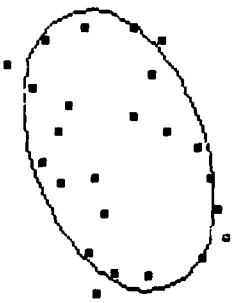
MAJOR AXIS = 128.70
MINOR AXIS = 44.07
ORIENTATION = 151.00
AREA = 5672.04



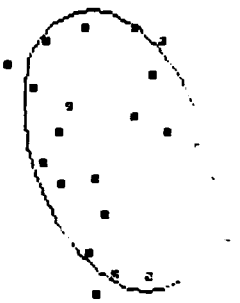
MAJOR AXIS = [illegible]
MINOR AXIS = [illegible]
ORIENTATION = [illegible]
AREA = [illegible]



MAJOR AXIS = 108.23
MINOR AXIS = 92.62
ORIENTATION = 142.00
AREA = 10023.92

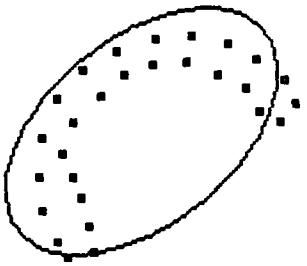


MAJOR AXIS = 99.80
MINOR AXIS = 59.49
ORIENTATION = 110.00
AREA = 5933.93

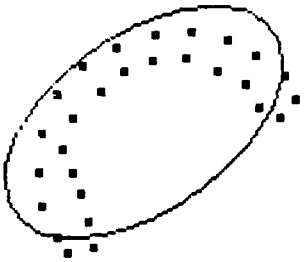


MAJOR AXIS = [illegible]
MINOR AXIS = [illegible]
ORIENTATION = [illegible]
AREA = [illegible]

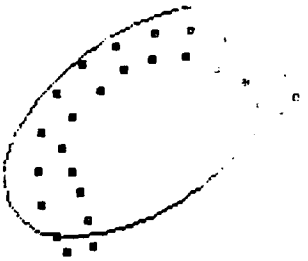
Fig. 3. Elliptical approximations preserving area, orientation, and aspect ratio of the bounding rectangles in Figure 2.



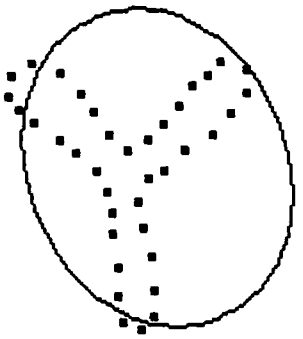
MAJOR AXIS = 108.07
MINOR AXIS = 63.37
ORIENTATION = 39.00
AREA = 6848.79



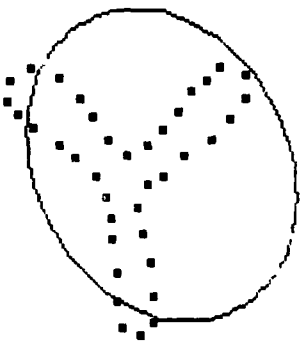
MAJOR AXIS = 107.81
MINOR AXIS = 59.97
ORIENTATION = 35.00
AREA = 6466.01



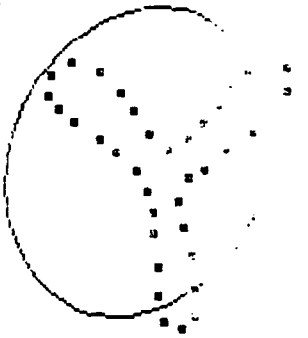
[illegible]OR AXIS = [illegible]31
[illegible]INOR AXIS = [illegible]47
[illegible]IENTATION = [illegible]30
[illegible]EA = [illegible]31



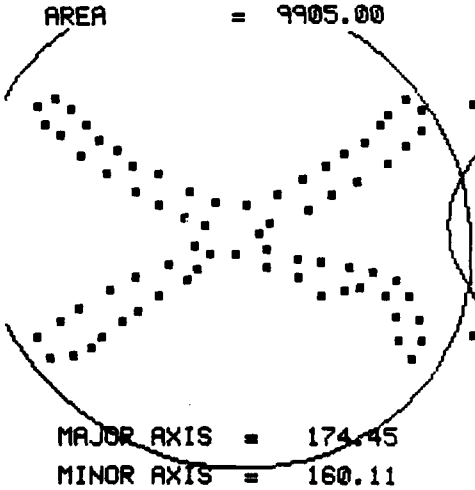
MAJOR AXIS = 111.23
MINOR AXIS = 89.05
ORIENTATION = 112.00
AREA = 9905.00



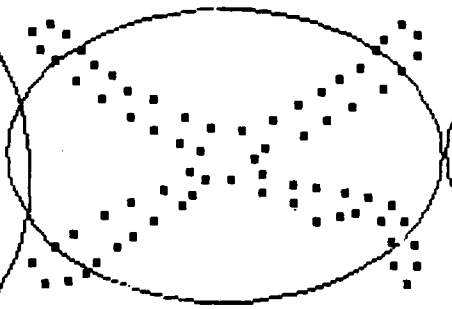
MAJOR AXIS = 110.84
MINOR AXIS = 86.81
ORIENTATION = 117.00
AREA = 9621.62



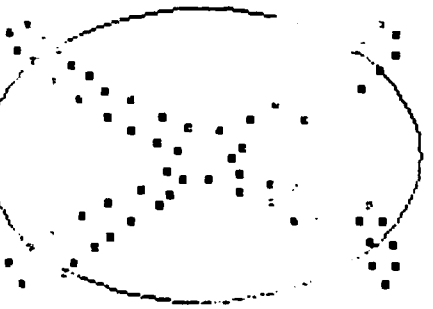
[illegible]OR AXIS = [illegible]17
[illegible]INOR AXIS = [illegible]97
[illegible]IENTATION = [illegible]00
[illegible]EA = [illegible]43



MAJOR AXIS = 174.45
MINOR AXIS = 160.11
ORIENTATION = 146.00
AREA = 27931.78



MAJOR AXIS = 149.00
MINOR AXIS = 100.00
ORIENTATION = .00
AREA = 14900.00



[illegible]OR AXIS = [illegible]00
[illegible]INOR AXIS = [illegible]00
[illegible]IENTATION = [illegible]00
[illegible]EA = [illegible].00

Fig. 3. Continued.

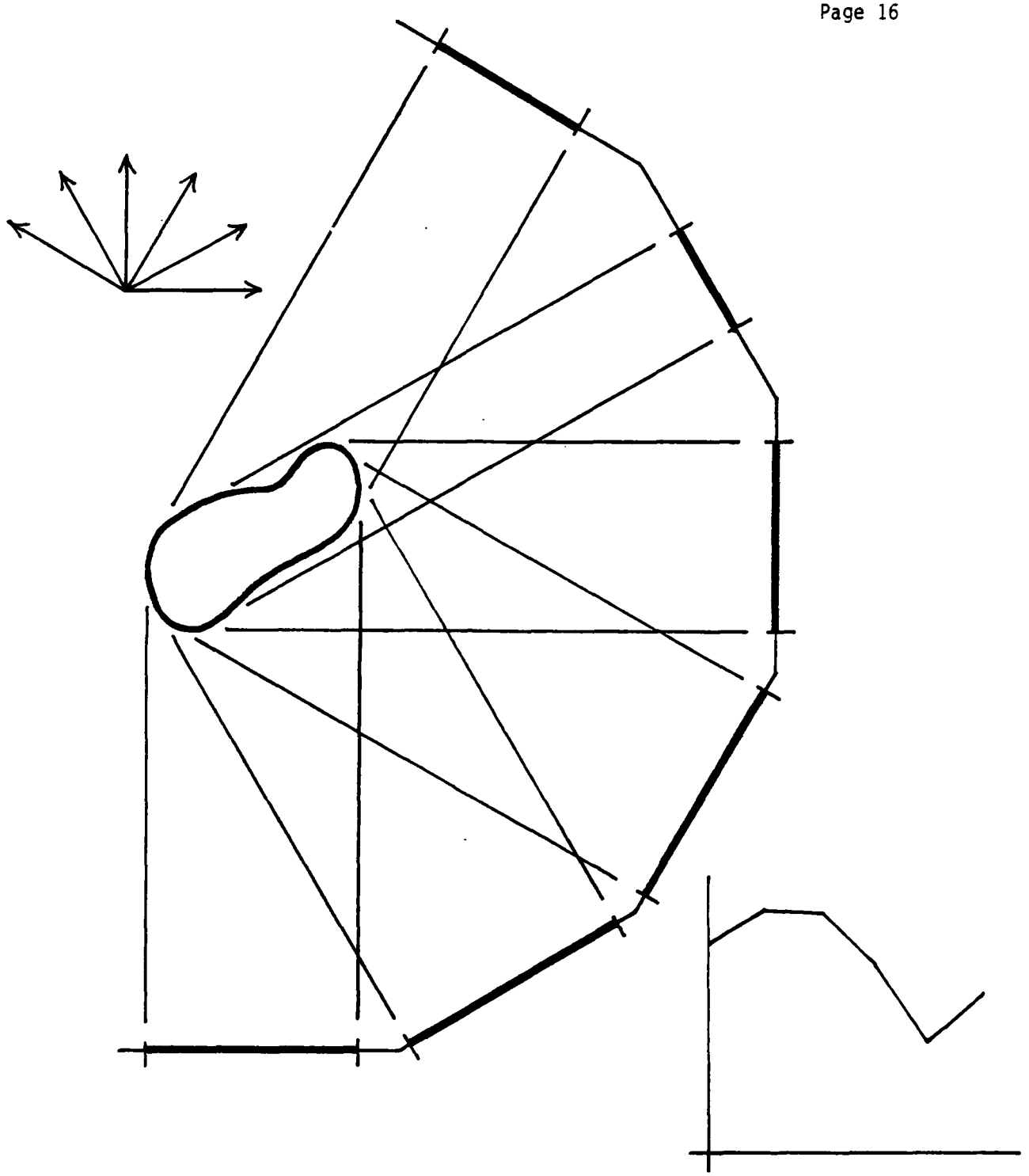


Fig. 4. The projection of a blob onto the k *viewing lines* corresponds to the dot product of each boundary point with k unit vectors. This gives $W(\Theta)$: the width of the blob's projection as a function of the orientation of the viewing line.

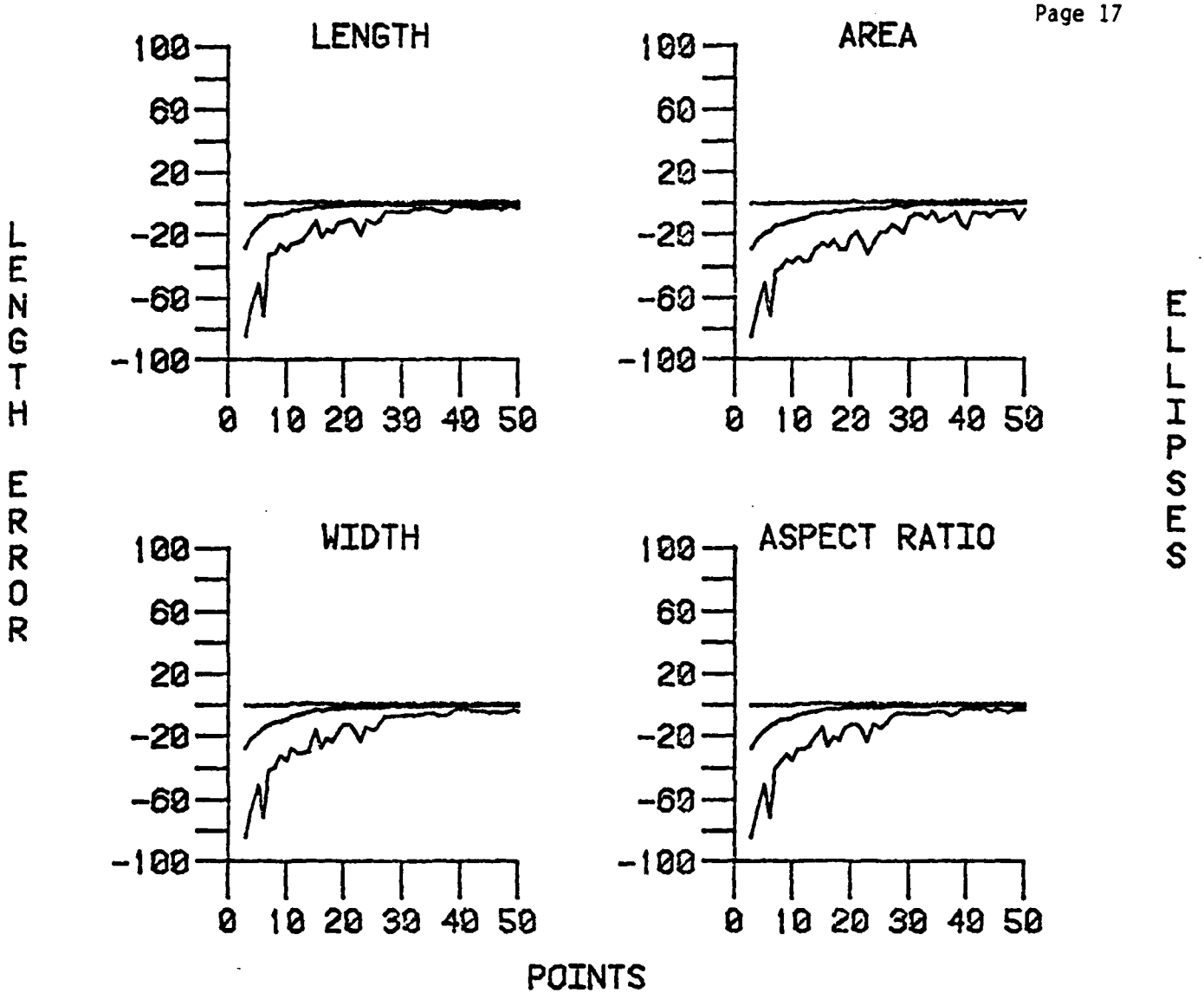
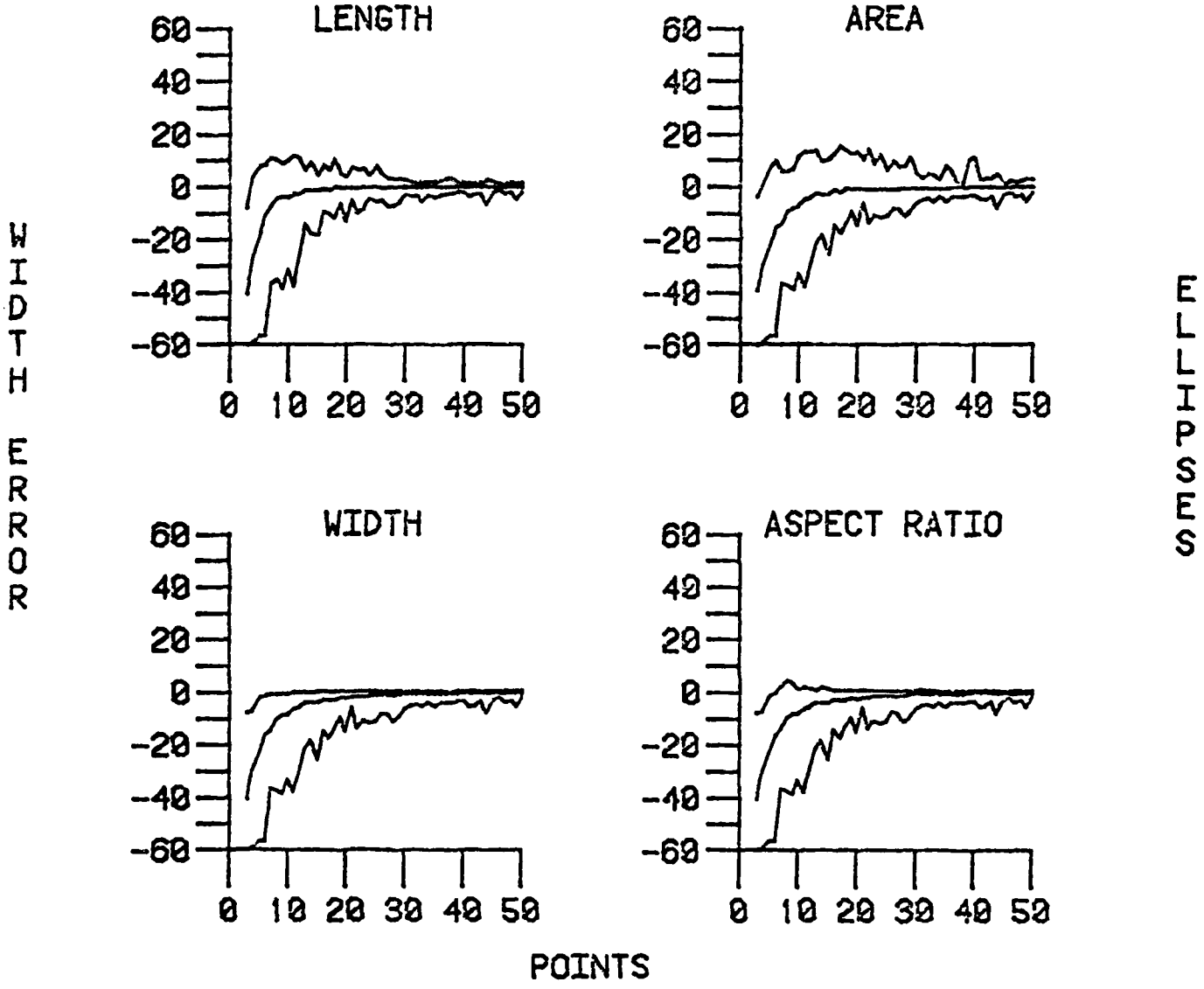
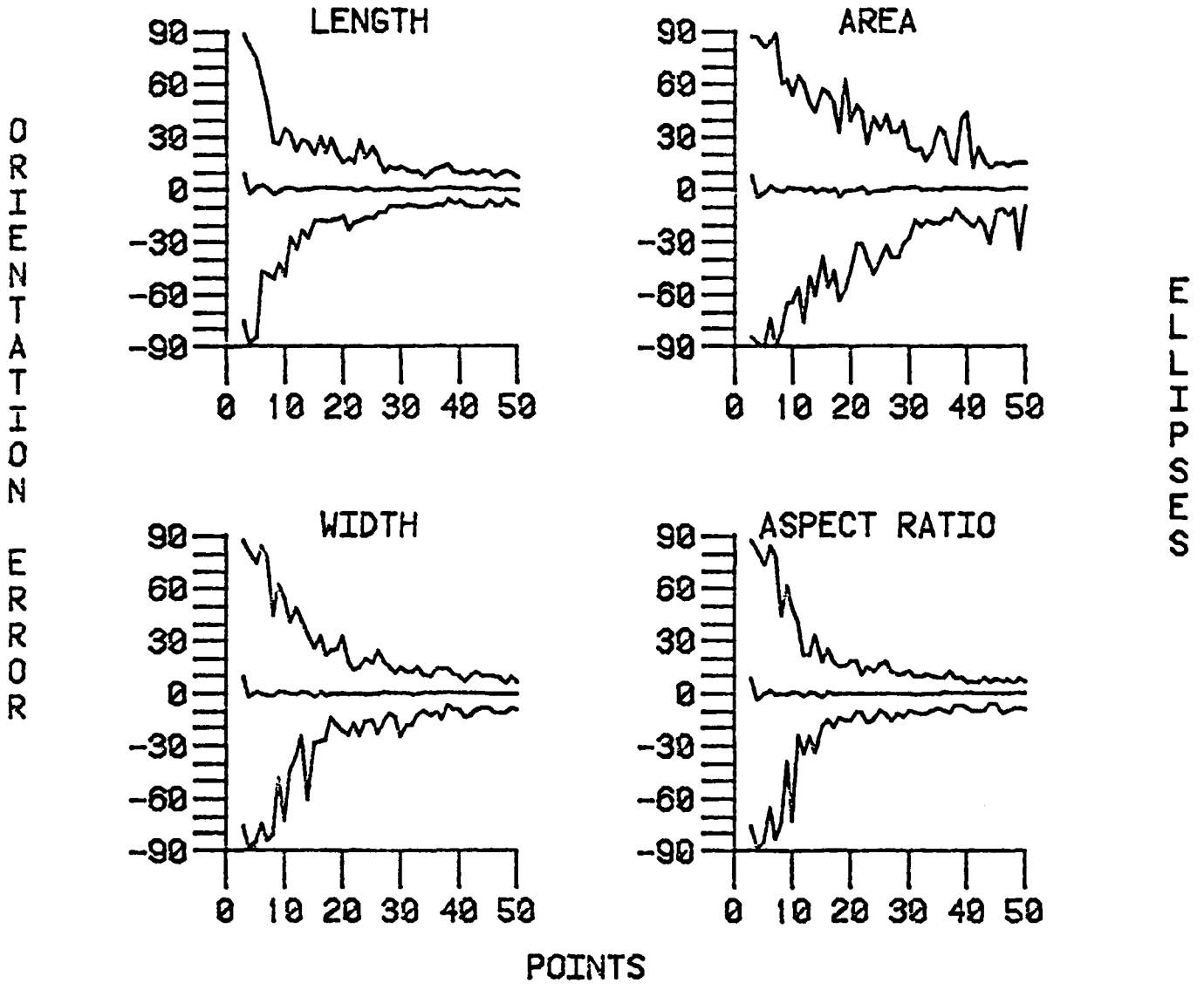


Fig. 5. a) Minimum, maximum, and mean error in the length estimate;
b) Minimum, maximum, and mean error in the width estimate;
c) Minimum, maximum, and mean error in the orientation estimate.


LENGTH
AREA
WIDTH
ASPECT RATIO
60
40
20
0
-20
-40
-60
0 10 20 30 40 50
WIDTH ERROR
ELLIPSES
POINTS


LENGTH
AREA
WIDTH
ASPECT RATIO
90
60
30
0
-30
-60
-90
0 10 20 30 40 50
ORIENTATION ERROR
ELLIPSES
POINTS

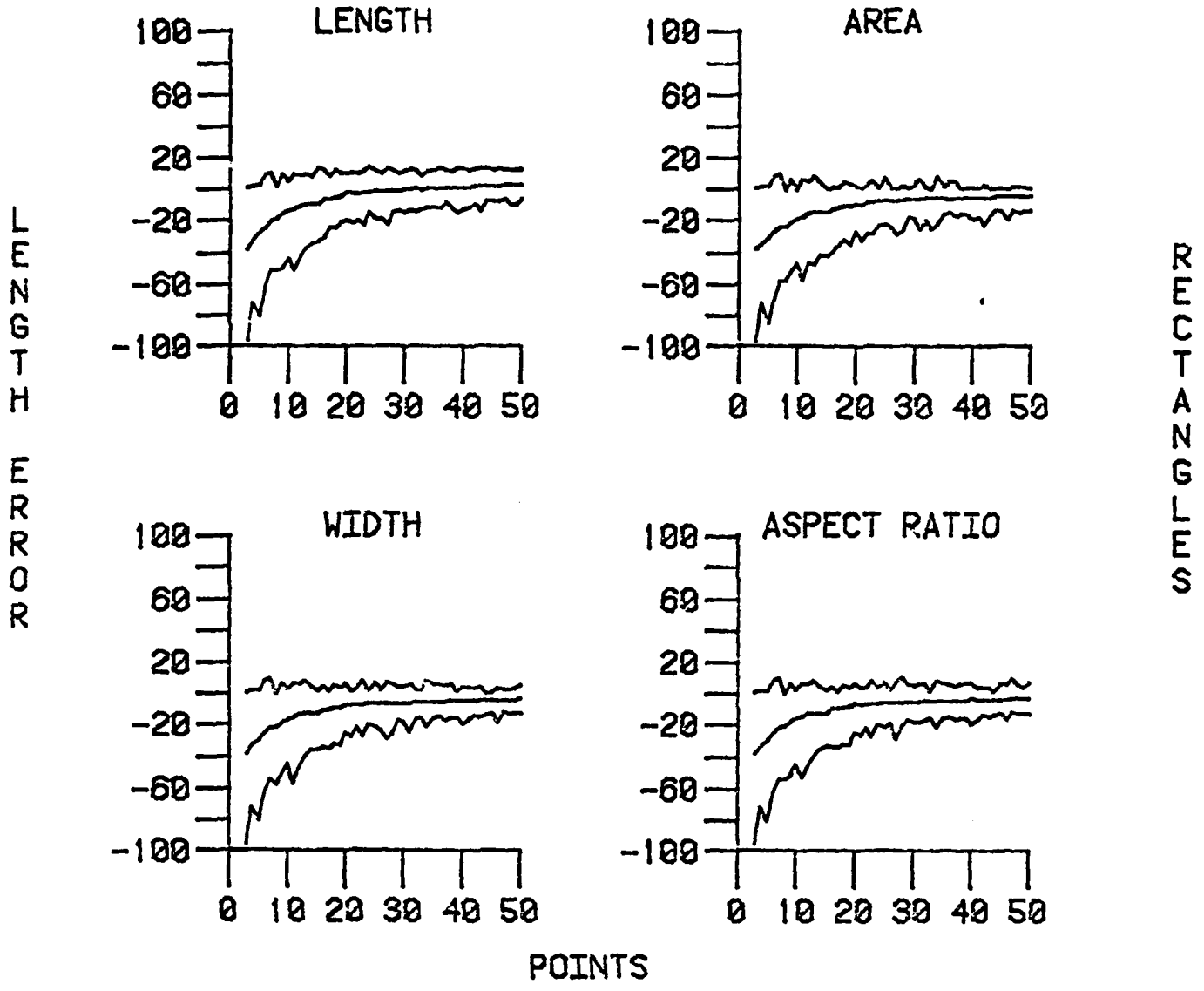
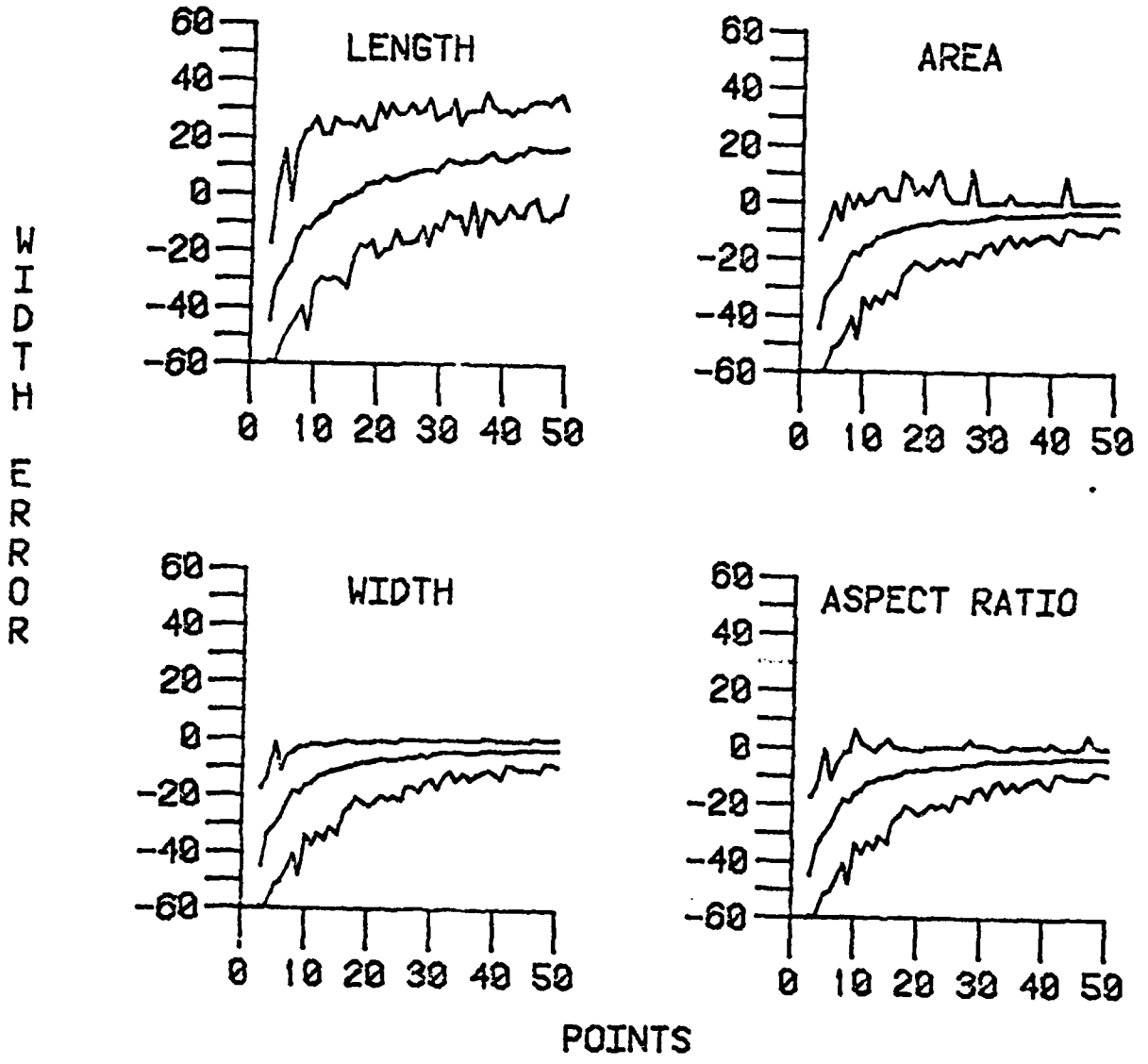
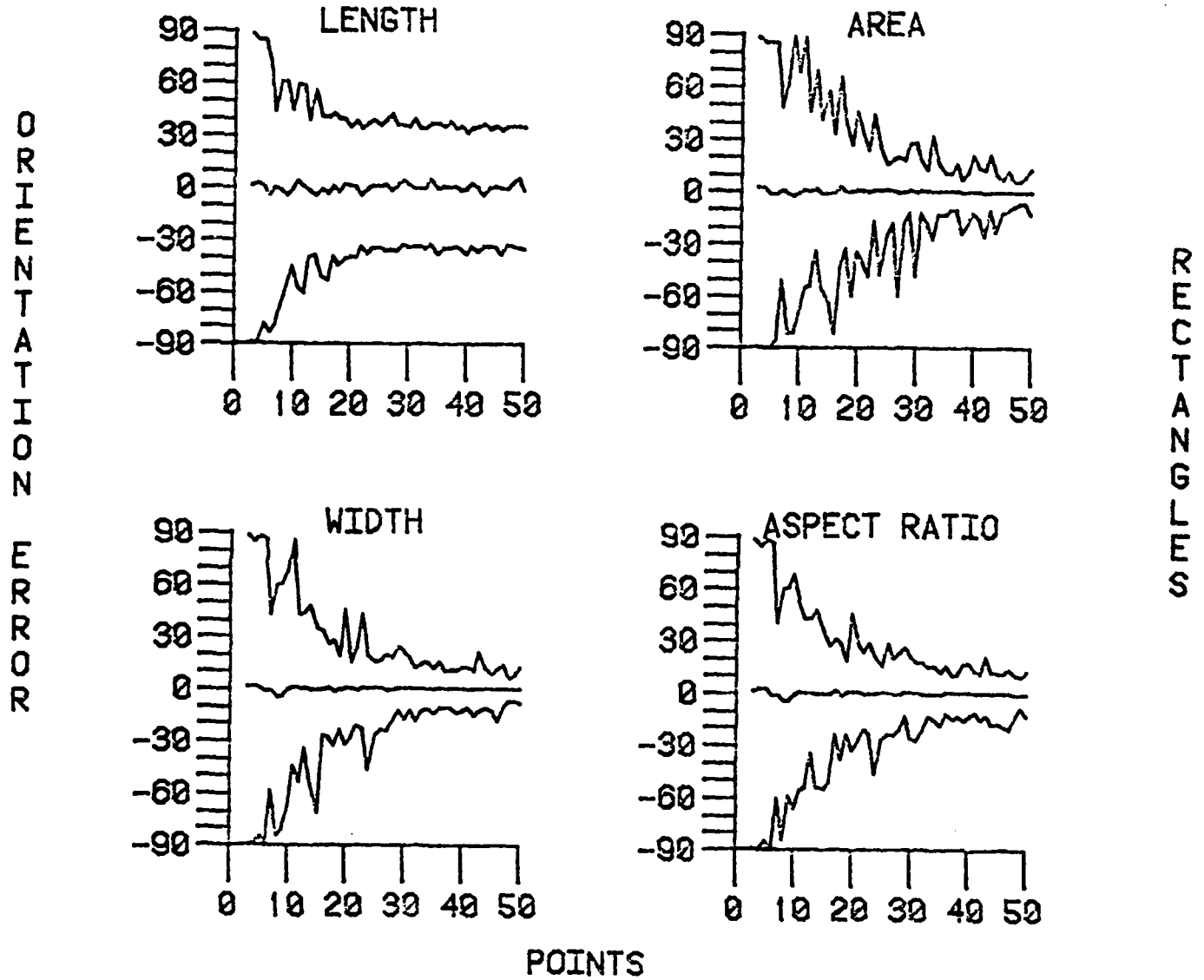


Fig. 6. a) Minimum, maximum, and mean error in the length estimate;
b) Minimum, maximum, and mean error in the width estimate;
c) Minimum, maximum, and mean error in the orientation estimate.


LENGTH
AREA
WIDTH
ASPECT RATIO
60
40
20
0
-20
-40
-60
0 10 20 30 40 50
WIDTH ERROR
RECTANGLES
POINTS


LENGTH
AREA
WIDTH
ASPECT RATIO
90
60
30
0
-30
-60
-90
0 10 20 30 40 50
ORIENTATION ERROR
RECTANGLES
POINTS